Université Nice Sophia Antipolis

Membre de UNIVERSITÉ CÔTE D'AZUR

POLYTECH NICE-SOPHIA

Département Sciences Informatiques

React

Programmation Web Client

RAPPORT DE PROJET INDIVIDUEL

(François Michaudon, Hugo Mallet)

KOFFI Paul

Identifiant GitHub : PaulKoffi

Sommaire

Tâches effectuées

- Authentification JWT
- Déploiement de l'application sur un serveur SSH distant disponible au Domaine suivant http://paulkoffi.fr
- Formulaire de contact avec envoi de courriel et validation des champs
- Affichage sous forme de table
- Gestion de la persistance du DarkMode et popup de confirmation si le Système de l'utilisateur change de thème (Clair/Sombre) afin D'appliquer celui-ci sur notre site
- Gestion du CSS et de la mise en place de la page principale de l'application (HomePage)

Stratégie employée pour la version des gestions avec git

Un branching continu

Le projet a été commencé sur la branche Back avec la conception du backend en nodeJS. Par la suite une branche front a été crée pour l'implémentation du frontend. Sur ce projet de cette envergure, nous avons préféré les choses simplement avec un système de branches minimales et pas de validation via Pull Request. Un conflit majeur dû à l'utilisation de bootstrapp de mon côté et de AngularMaterial par un autre membre du groupe m'a conduit à la création de la branche develop pour le fix et ensuite à la branche « lab » où j'ai développé personnellement jusqu'à la fin du projet. Etant en charge de la structure et de l'intégration des différents composants (SIDEBAR, TABLE, MAP, GRAPHE), cette branche a finalement servi de point de convergence.

Une release va être effectué pour la soumission finale du sujet et va donc constituer la première version de l'application.

Solutions choisies

- Implémentation du login : La solution choisie fut l'authentification JWT. Ce protocole facile à implémenter est d'autant plus efficace qu'une authentification normale. Le backend génère un token qui sert à authentifier directement l'utilisateur pour une prochaine session. Ce token arrive à expiration au bout du temps paramétré par le développeur.

- Implémentation du Dark mode : Pour l'implémentation du DarkMode, j'ai utilisé une combinaison du ThemeContext existant déjà dans la solution du projet iTunes avec une modification via le local Storage pour faire perdurer l'état même après fermeture du navigateur ou déconnexion.

  Concernant la gestion proprement dite du thème de l'application, j'ai opté pour une solution facile qui consiste à ajouter un élément de style en testant directement la valeur du thème depuis le Context. Une meilleure stratégie aurait été d'utiliser un ThemeProvider ce qui permet d'impacter le style général sans le faire sur chaque élément ou groupe d'éléments. Cela aurait été moins fastidieux.

  Une autre option si on n'avait pas eu l'obligation de demander la confirmation à l'utilisateur avant de passer dans un thème DARK, aurait été de mettre tout ce qui concerne le style DARK de notre application avec une balise **(prefers-color-scheme : dark)**.

  Pour le popup de confirmation, j'ai tirer partie de la librairie (https://www.npmjs.com/package/react-confirm-alert) qui permet d'ouvrir une boite de dialogue que tu peux placer sur l'écran comme bon nous semble. Son utilisation est assez facile car elle permet d'exécuter le bout de code voulue en fonction du choix de l'utilisateur. Une alternative aurait été peut-être l'utilisation d'un modal Bootstrap mais serait plus complexe à utiliser.

- Formulaire de login & envoi de Mail : J'ai utilisé le package react-bootstrap pour le formulaire de Login et d'envoi de Mail. Pour la validation des champs spécifiques comme l'adresse Email, j'ai tiré profit de la validation faîte par Bootstrap. L'inconvénient c'est que le message d'erreur n'est pas aussi stylisé qu'avec la magnifique librairie https://www.npmjs.com/package/react-jsonschema-form-validation.

  Pour l'envoi des notifications (Authentification Failure/ Mail sending), j'ai utiliser la superbe librairie (https://fkhadra.github.io/react-toastify/introduction/) qui permets d'avoir des alertes personnalisables, qui apparaissent pendant un temps passé en paramètre et que l'utilisateur a bien sûr le choix de faire disparaitre sans attendre le temps imparti.

Difficultés rencontrées

Les principales difficultés sont venues du linter du projet qui était assez stricte. Pour moi qui suis habitué à développer souvent avec Angular, je n'avais jamais fait du react auparavant. J'ai trouver l'architecture de REACT assez léger comparé à la rigueur qu'imposait Angular. J'ai eut un peu de mal à m'y retrouver au début et j'ai ensuite fait une architecture de notre projet mieux organisé.

Une autre difficulté mais mineure fut l'utilisation de javascript en front à la place du typescript alors que j'en avais l'habitude.

Temps de développement / tâche

- Authentification JWT : 1h
- Déploiement de l'application sur un serveur SSH distant disponible au Domaine suivant http://paulkoffi.fr 1h
- Formulaire de contact avec envoi de courriel et validation des champs : 30 min
- Affichage sous forme de table : 2H
- Gestion de la persistance du DarkMode et popup de confirmation si le Système de l'utilisateur change de thème (Clair/Sombre) afin D'appliquer celui-ci sur notre site : 1 jour

- Gestion du CSS et de la mise en place de la page principale de l'application (HomePage) (sur tout le projet)

Codes

Optimal: (https://github.com/wak-nda/prog-web-ADKN/blob/main/frontend/src/pages/Login.js)

Cette page de login me semble plutôt complet, simple et optimal. Si l'utilisateur est déjà connecté, nous avons une redirection automatique vers la page principale de l'application. La fonction de validation du formulaire s'assure que l'utilisateur a bien remplis tous les champs afin de dégriser le bouton de connexion. La précision du type courriel permet d'avoir une adresse mail syntaxiquement correcte. Enfin lorsque le formulaire est validé, nous faisons appel au service de login pour connecter l'utilisateur, étant une promesse nous n'oublions pas le mot clé **await** pour attendre la réponse du backend. Si l'authentification est réussie, l'utilisateur est redirigé vers la page principale de l'application. Dans le cas contraire, une superbe alerte est envoyée à l'utilisateur pour lui signaler l'échec de la connexion de son compte utilisateur.

Amélioration : (https://github.com/wak-nda/prog-web-ADKN/blob/main/frontend/src/services/api.js)

Cette fonction permet simplement de choisir en fonction de l'environnement de développement la première partie de l'adresse du serveur backend. En environnement de production, il pingue le backend déployé sur mon serveur ssh au nom de domaine paulkoffi.com. Elle peut être amélioré en mettant principalement toutes les routes de l'application qui seront donc dans un fichier constant.js par exemple.